A Project Report on

HAND GESTURE RECOGNITION AND CONTROL

Submitted by

PRITHVI RAM.G (115004146)

RAMASUBRAMANIAN.J (115004157)

SANTHANA VIKRAMA RAJAVARMAN.M (115004168)

In partial fulfillment for the award of the degree of

Bachelor of Technology

In

Electronics and Communication Engineering

School of Electrical & Electronics Engineering
SASTRA University
Thanjavur, India – 613 401

April, 2015

BONAFIDE CERTIFICATE

Certified that the project work entitled **"HAND GESTURE RECOGNITION AND CONTROL"** submitted to SASTRA University, Thanjavur by PRITHVI RAM.G (Reg.No:115004146), RAMASUBRAMANIAN.J (Reg.No:115004157), SANTHANA VIKRAMA RAJAVARMAN.M (Reg. No:115004168), in partial fulfillment for the award of the degree of Bachelor of Technology in Electronics and Communication Engineering is the work carried out independently under my guidance during the period Dec 2014 – April 2015.

Project Guide

[NARASIMHAN.K]
[AP III, ECE, SASTRA University]

External Examiner **Internal Examiner**

Submitted for the University Exam held on _____

DECLARATION

We submit this project work entitled "**HAND GESTURE RECOGNITION AND CONTROL"** to SASTRA University, Thanjavur in partial fulfillment of the requirements for the award of the degree of "**Bachelor of Technology**" in **"Electronics and Communication Engineering".** We declare that it was carried out independently by us under the guidance of NARASIMHAN.K, AP III, ECE, SASTRA University.

PRITHVI RAM.G	**(Reg. No: 115004146)**
RAMASUBRAMANIAN.J	**(Reg. No: 115004157)**
SANTHANA VIKRAMA RAJAVARMAN.M	**(Reg. No: 115004168)**

Date: Signature: 1.

Place:

2.

3.

ACKNOWLEDGEMENTS

First of all I express my gratitude to **Prof. R. Sethuraman**, Vice Chancellor, SASTRA University who provided all facilities and the necessary encouragement during the course of study. I extend my sincere thanks to **Dr. G. Bhalachandran**, Registrar, SASTRA University for providing the opportunity to pursue this project.

I dedicate my wholehearted thanks to **Dr. B. Viswanathan**, Dean, SEEE, SASTRA University and **Prof. M. Narayanan,** Dean-Student affairs, SEEE, SASTRA University, for their moral support. I also thank **Dr. K. Thenmozhi,** Associate Dean (ECE) who motivated me during the project work.

I am greatly indebted expressing my gratitude to my project guide **Mr. K. Narasimhan,** AP III, ECE, SASTRA University for his continuous support and guidance throughout the process during the pursuit of my project work. His deep insight in the field and invaluable suggestions helped me in making progress through my project work.

I wish to thank all my teaching and non-teaching staff members of the Department of Electronics and Communication Engineering of SEEE for their support and cooperation. I also extend my gratitude to all teachers who taught me since my childhood days.

I would like to dedicate this work with tremendous love to my Parents for their unlimited, ultra-supportive, encouraging, sacrifices and unconditional love throughout my entire life.

I am also greatly indebted to my friends for their invaluable suggestions, for their assistance during the testing of our project and also for their tremendous moral support during tough times. They deserve much love and thanks.

Above all, I thank the Almighty for the Divine Grace I received throughout the research work, without which I could not have completed this work.

TABLE OF CONTENTS

TITLE	PAGE NO.
CHAPTER – 1: INTRODUCTION	1
1.1 Project Justification	1
1.2 Project Goals	1
1.3 Scope of Project	2
CHAPTER – 2: METHODOLOGY	3
2.1 General Block Diagram	3
2.2 Image Pre-Processing	4
2.3 Feature Extraction	8
2.4 Training Stage	9
2.5 Testing Stage	11
2.6 Hardware Interfacing Stage	12
CHAPTER – 3: BACKGROUND THEORY	14
3.1 Haar Cascade Classifier	14
3.2 YCbCr Color Space	16
3.3 Dilation and Erosion	17
3.4 Connected Component Labelling	18
3.5 Contour and Convex hull	19
3.6 Orientation Calculation	20
3.7 Hu Moments	21
3.8 SVM classifier	22
CHAPTER – 4: SYSTEM REQUIREMENTS & INSTALLATION	24

PROCEDURES	
4.1 System Requirements	24
4.2 Python Installation	24
4.3 Open CV-Python Binding	24
4.4 Camera Specifications	25
4.5 Raspberry PI	26
4.5.1 Specifications	27
4.5.2 Raspbian OS installation	27
4.5.3 GPIO Header	28
4.6 Configuration of Ethernet and Wi-Fi	29
4.7 Software Required	30
4.8 Remote Access for Raspberry Pi	31
4.9 Motor Driver IC	32
4.10 DC motor specifications	33
4.11 Power Sources	33
CHAPTER – 5: RESULT AND ANALYSIS	
5.1 Skin color detection	34
5.1.1 RGB color space	35
5.1.2 HSV color space	36
5.1.3 YCbCr color space	37
5.2 Illumination condition and background environment	38
5.3 Output	39
CHAPTER – 6: CONCLUSION AND FUTURE RECOMMENDATION	40

6.1 Conclusion	40
6.2 Future recommendation	40
REFERENCES	41

LIST OF FIGURES

S.No.	No.	FIGURE NAME	PAGE No.
1	2.1	General Block Diagram of the system	3
2	2.2	Block diagram for Image Pre-Processing	4
3	2.3	Images captured by webcam for different hand postures	5
4	2.4	Face Subtracted images for different hand postures	5
5	2.5	Skin color segmented region in YCbCr Color space and image after closing operation	6
6	2.6	Selecting Hand region by using connected components labelling	7
7	2.7	Contour of the hand region in the image	7
8	2.8	SVM Classifier Training stage Block Diagram	10
9	2.9	Block diagram for Implementation stage	11
10	2.10	Complete circuitry of Hardware unit	12
11	2.11	Bot connected with Raspberry Pi board through motor driver IC	13

12	3.1	Finding the sum of the shaded rectangular area	15
13	3.2 a	Edge features	15
14	3.2 b	Corner features	15
15	3.2 c	Line features	16
16	3.3	Chrominance plane for Y = 0, 128 and 255	16
17	3.4	Image before and after Closing operation	18
18	3.5	Connected Component Labelling	19
19	3.6	Contour of the hand region and its convex hull	20
20	3.7	Orientation of hand region with respect to vertical axis	21
21	3.8	Linear SVM Classifier	23
22	4.1	Raspberry Pi Model B+ View	26
23	4.2	Raspberry Pi Model B+ GPIO Header	28
24	4.3	Wireless Network Connection Properties	30
25	4.4	L293D Motor Driver IC	32
26	5.1	Sample image used for skin color detection and analysis	34
27	5.2	Skin color detection in RGB color space	35
28	5.3	Skin color detection in HSV color space	36
29	5.4	Skin color detection in YCbCr color space	37
30	5.5	False detection of hand region	38

31	5.6	Final output	39

LIST OF TABLES

S.No	No.	TABLE NAME	PAGE No.
1	2.1	Feature vector table for two different hand gestures	9
2	2.2	Hu-moments of the hand region for two different hand posture	9
3	2.3	Confusion matrix of SVM Classifier	10

LIST OF ABBREVIATIONS

HCI	Human Computer Interaction
SVM	Support Vector Machine
GPIO	General Purpose Input Output
RGB	Red Green Blue
HSV	Hue Saturation Value

ABSTRACT

With the development of technology, computers are playing crucial role in our day-to-day life. Like humanity, computers are also progressing through various revolutions. Current generation computers are so advanced that the traditional user interaction devices like keyboard and mouse do not meet the challenges in many cases like Human-Robot interaction. So, there is a need to revolutionize the interactivity of humans with computers. One of the best solutions is the use of human hands as the input device. Because of the vast degree of freedom, it provides natural interaction with computers. A gesture recognition is a field with the goal of interpreting human gestures via mathematical algorithm and it is a way that the computers to begin to understand human body language. This technology has various applications, both in commercial and industrial fields.

In this project, we propose a vision based control system using hand gestures captured from the camera. The system allows the user to give various commands such as start, stop, moving forward, etc. remotely to an external automobile unit using simple hand gestures. It recognizes up to four different hand gestures which are interpreted as commands for driving the automobile unit. The reliability and efficiency of this application depend on the illumination conditions and background

There are three stages in our system. In the first stage, dataset of the desired hand gestures is captured in a uniform environment and they are used to train the Support Vector Machine (SVM) classifier. The second stage captures the real time video feed from the webcam on the computer, preprocesses every frame to get the region of interest and finally recognizes the hand posture by the trained SVM classifier. In the final stage, the recognized hand postures are converted to suitable commands and are transmitted to a remote device to control the automobile unit.

Chapter 1

1. Introduction

1.1 Project Justification

The interaction between human being and the computer is called as Human Computer Interaction. The most widely used input devices are mouse and keyboard. But the growth of technology in the field of computers is tremendously over the past few years. Hence, to cope with the technological advancement, we need to find replacement for these conventional input devices. One of the methods is the use of hand gestures as the input to the system. This method is more effective than the former method because it provides a natural interaction. Nowadays, most of the laptop and desktop computers are coming with pre-installed webcam. Hence there is no need of any external hardware module. In order to fully utilize the potential of a webcam, it can be used for vision based human computer interaction. This would effectively reduce the need for a mouse for basic application such as motion controller. Many extensive researches are going towards developing novel devices for interaction using hand gestures in the field of computer games, motion control and robotics.

1.2 Project Goals

- To develop a real time system for interaction with an application.
- To develop a prototype model for detecting and recognizing hand gestures.
- Building a grammar that generates gesture commands.
- To control a hardware unit based on the gesture commands.

1.3 Scope of Project

Generally there are two approaches for HCI using hand gestures. One method makes use of hardware unit such as Data Gloves or colored markers. In this method, the user must wear the hardware device which reduces the user convenience. Another method is a vision based system, which makes use of image processing techniques to recognize human gestures. This method does not need any specific hardware device like gloves, etc. In our system, we proposed vision based method to perform the hand gesture recognition. The scope of this project is to remotely control an automobile unit using human hand gestures.

Chapter 2

2. Methodology

In this chapter, we explained about the various methods and strategies used in the design and development of the hand gesture recognition system. The atomic modules specified in this chapter are explained in Chapter 3.

2.1 General Block Diagram:

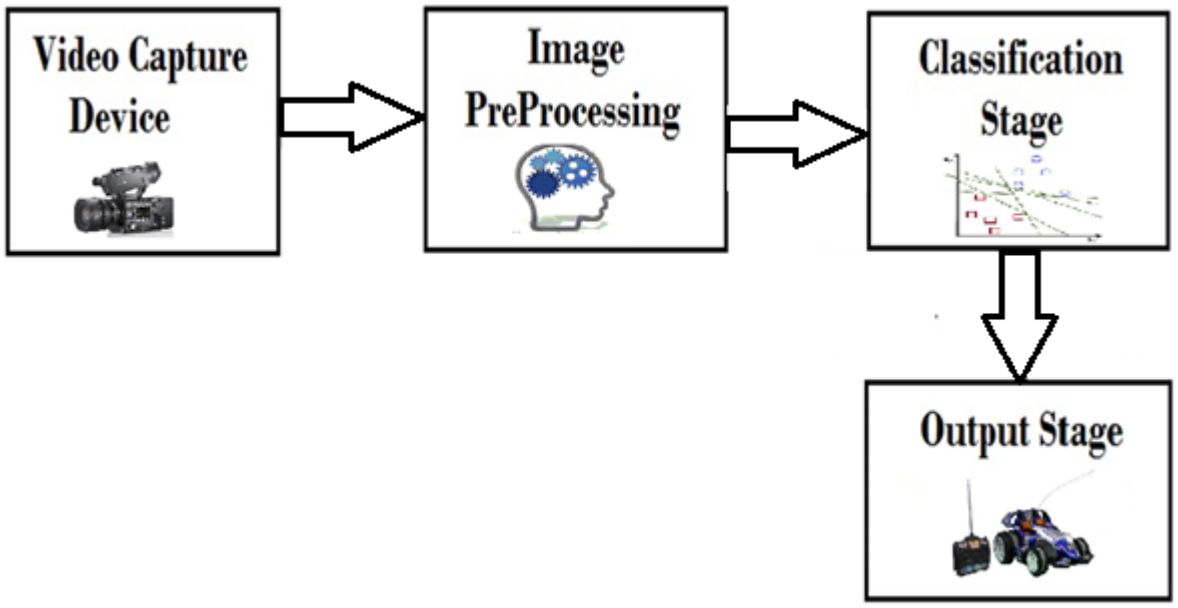

Fig 2.1 General Block Diagram of the system

As shown in Figure 2.1 the images from video capture device are pre-processed before classification which involves two levels training and testing. The image pre-processing is common for both classification levels. Each blocks in the general block diagram are explained in this chapter.

2.2 Image Pre-Processing

The following block diagram shows the various stages involved in the image pre-processing.

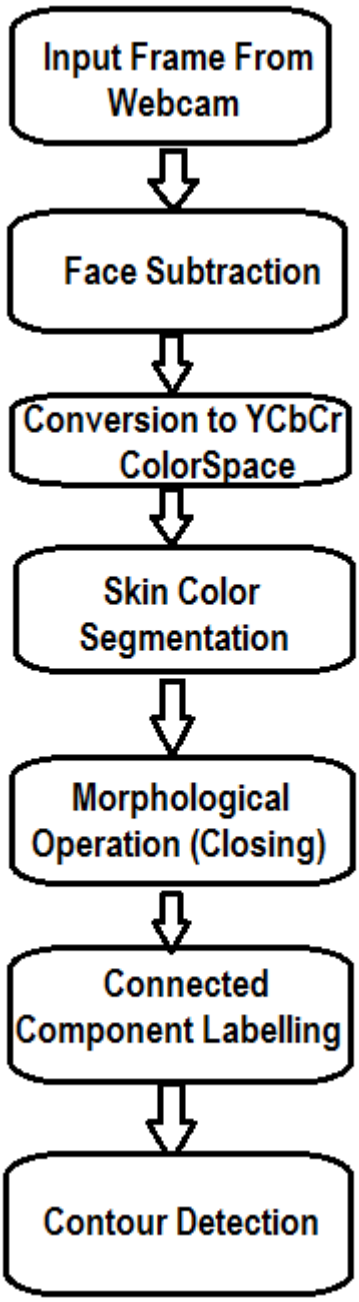

Fig 2.2 Block diagram for Image Pre-Processing

Each frame from the video stream is pre-processed before extracting feature of the Hand gesture both in testing and training (if training images are not collected in special environment). Two hand postures used in this project are shown in Fig 2.3

Fig 2.3 Images captured by webcam for different hand postures

Initially the Faces in each frame are subtracted. For this the face region is first detected using Haar cascade classifier and the Face detected region is subtracted from the original image by simply substituting black color value in that region as shown in Fig 2.4

Fig 2.4: Face Subtracted images for different hand postures

Face subtraction will reduce the chance of false detection of face as hand, as color value of face is same as color value of hand. After the Face subtraction is over the image is color threshold to get the skin color area only. This skin color segmentation can be done in different color spaces such as RGB, HSV and YCbCr. The choice of color space for skin color for covering the front and back side of the hand above wrist area depends on the illumination condition. In our project we have used YCbCr color space, skin color region in the Fig 2.4 is identified as shown in the Fig 2.5. The skin color range in YCbCr color space is (135,183) for Cr, (120,154) for Cb and Y above 60. Once the skin color area is selected, we have to do some morphological operation on it.

Fig 2.5Skin color segmented region in YCbCr Color space and image after closing operation

As the presence of noise and shadows in image, skin color area will not be smooth there might be presence of dots and gaps, in order to close those dots and gaps closing operation is needed. The kernel that we have used for closing is 5x5 unit matrix for erosion and 9x9 unit matrix for dilation. The result of closing operation on Fig 2.5 can be seen in Fig 2.6.

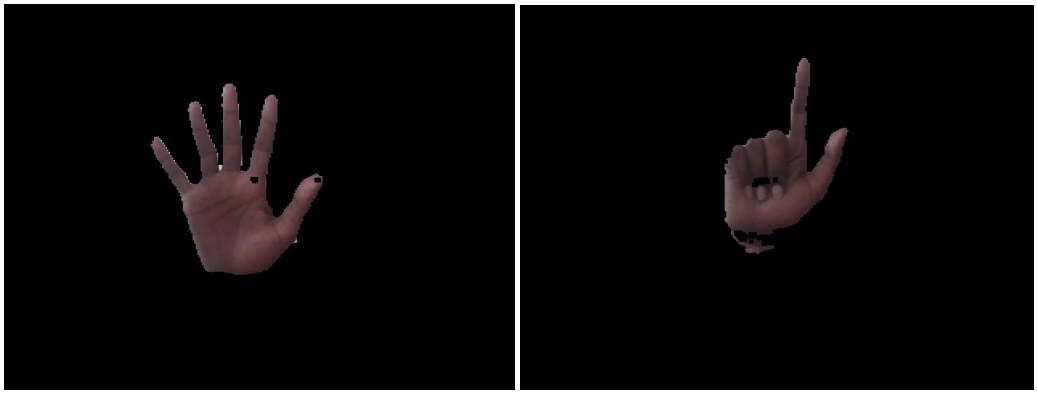

Fig 2.6 Selecting Hand region by using connected components labelling

Then the hand region is selected from the skin color segmented image by connected components labelling where each separate island in the images are labelled and the labelled component which has the highest probability of becoming hand region is chosen for further processing. The region with area above 10000 pixels and match closely with the template of the palm is considered as hand region. As shown in Fig 2.7, hand region alone will be selected by connected components labelling.

Fig 2.7 Contour of the hand region in the image

2.3 Feature Extraction

The contour is detected for the hand region, as shown in Figure 2.7 and the features are extracted from the contour calculated. In our project, we have used twelve different features of the contour which are as listed below,

- No of defect points
- Auxiliary ratio
- Contour area to Bounding rectangle area ratio
- Hull area to Bounding rectangle area ratio
- Solidity
- Hu-moments (7)

The features are explained as follows:

No of Defect Points: Defect points are the farthest points between two Hull points, which are the points present in the outer layer of the contour.

Auxiliary Ratio: Width to height ratio of the bounding rectangle, which is the minimum area rectangle covering the contour.

R/C Ratio: Ratio of area of the bounding rectangle and area covered by the contour.

R/H Ratio: Ratio of area of the bounding rectangle and area covered by the hull of the contour.

Solidity: Ratio of area covered by the contour and hull area.

Hu-moments: Hu-moments are the scale and rotational invariant parameters of a shape described using the image moments.

The feature values for different hand gestures are taken and are tabled as follows. These values are used to classify the gestures with the help of SVM classifier.

Table 2.1 Feature vector table for two different hand gestures

	No of Defects	Auxiliary ratio	R/C ratio	R/H ratio	Solidity
Palm	4	0.972	2.04	1.43	0.68
One	2	0.58	1.95	1.6	0.817

Table 2.2 Hu-moments of the hand region for two different hand posture

	HM1	HM2	HM3	HM4	HM5	HM6	HM7
Palm	0.000994	6.02E-08	2.33E-10	2.49E-10	5.7E-20	5.56E-14	1.91E-20
One	0.00087	0.00000023	1.66E-10	4.11E-11	3.25E-21	1.17E-14	-1.05E-21

2.4 Training Stage

In the training stage, desired hand postures are captured in convenient environment and features are extracted after pre-processing as mentioned above. As SVM classifier involves supervised learning, the response vector for the extracted features has to be manually set. Feature and response vector together are used to train the SVM classifier.

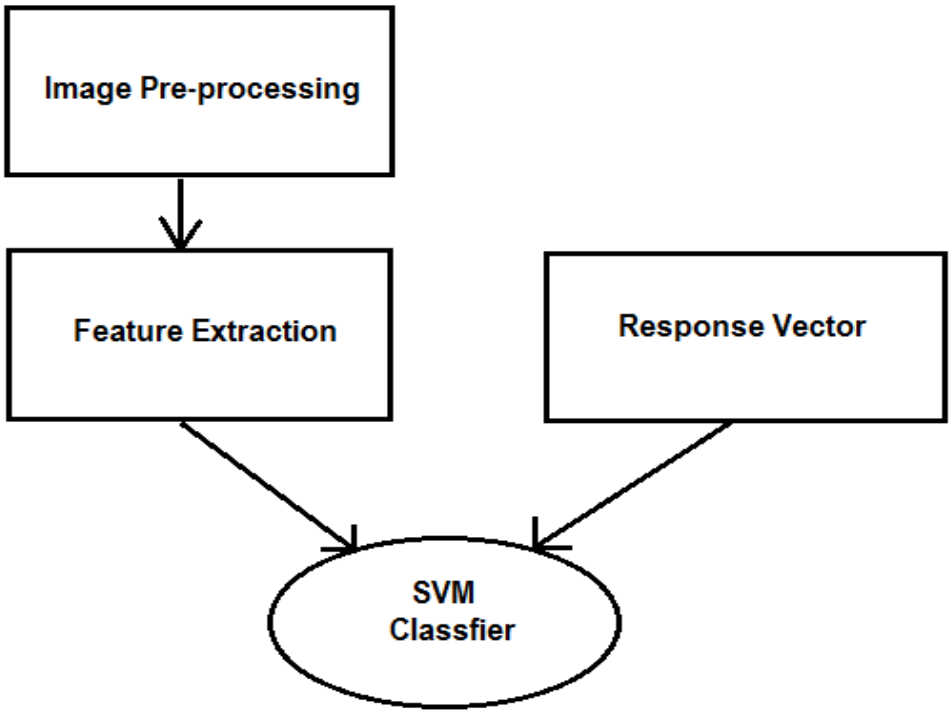

Fig 2.8SVM Classifier Training stage Block Diagram

The trained SVM classifier is tested for its accuracy by passing the same feature vector used in training for prediction. The confusion matrix of our SVM classifier for given 30 testing image set is shown below

Table 2.3 Confusion matrix of SVM Classifier

0	30
2	28

2.5 Testing Stage:

In the testing stage, images are captured from Webcam and each frame is pre-processed and features are extracted from it. The feature vector is passed to the SVM classifier for classification. Once the labelling is done by SVM classifier, Orientation of the hand posture is calculated.

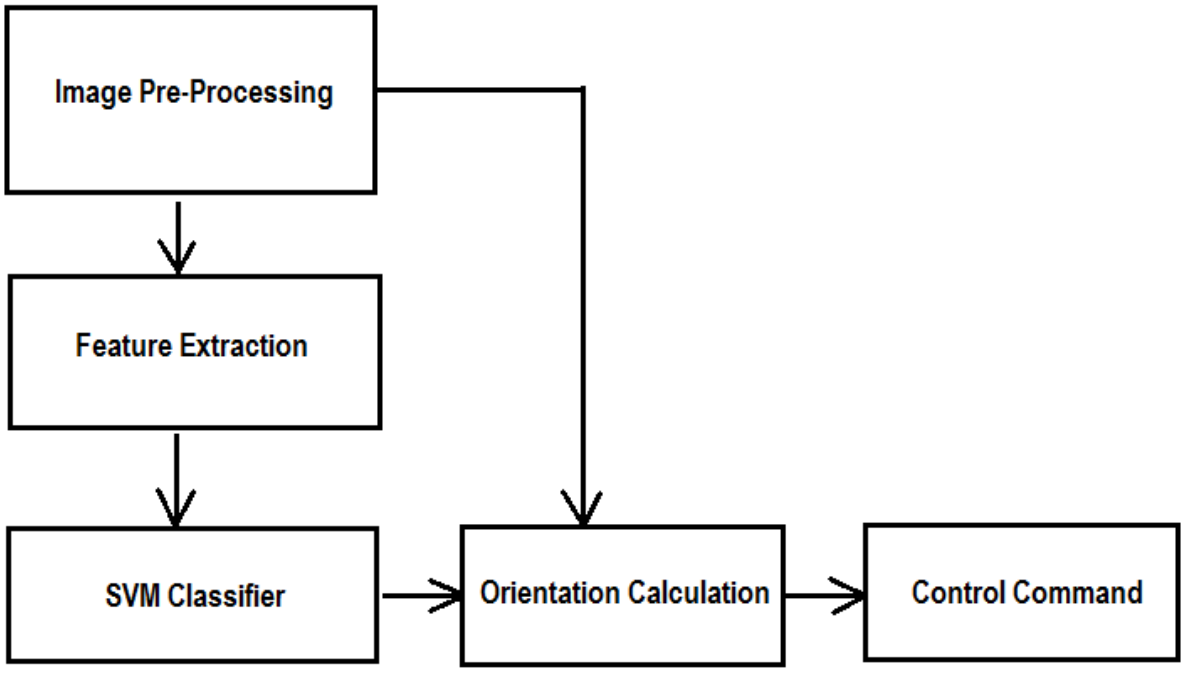

Fig 2.9 Block diagram for Implementation stage

Control command generated depends on the result of SVM classifier and orientation. The angles are calculated with respect to vertical axis which is considered as $\pm 90^{o}$, inclination to the left of vertical axis is measured as negative and to the right is measured as positive. In this project four commands are used, which are as follows

- Forward Straight: If the SVM classifier output is 1 and the orientation is between -60^{o} to 60^{o}.

- Forward Left: If the SVM classifier output is 1 and the orientation is less than -60º.
- Forward Right: If the SVM classifier output is 1 and the orientation is greater than 60º.
- Reverse: If the SVM classifier output is 2.
- Stop: If desired hand posture is not available in the image.

2.6 Hardware Interfacing Stage:

The control command generated is transmitted to the remote device which is the controlling unit used to control the bot. In our project we have used Raspberry-Pi as control unit. The received command is processed by the Raspberry-Pi to generate the corresponding signals in GPIO pins which are connected to the motor driver IC to control the motors used in the hardware unit. The Transmission of command is done through a Wireless Local Area Connection (WLAN) established between the system and Raspberry-Pi (ref. Chapter 4 for configuring Pi for WLAN connection).

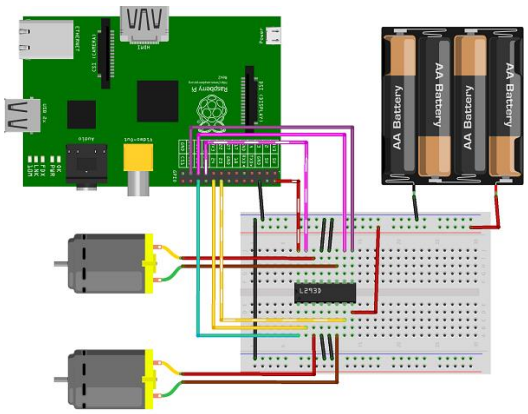

Fig 2.10 Complete circuitry of Hardware unit

In this project, the laptop acts as server, this is broadcasted to accept request from any client. And the client side python script is loaded in Raspberry Pi, which will request the server to open the socket to establish a connection between them. Once the connection is established, the server side program will send the control command to the client.

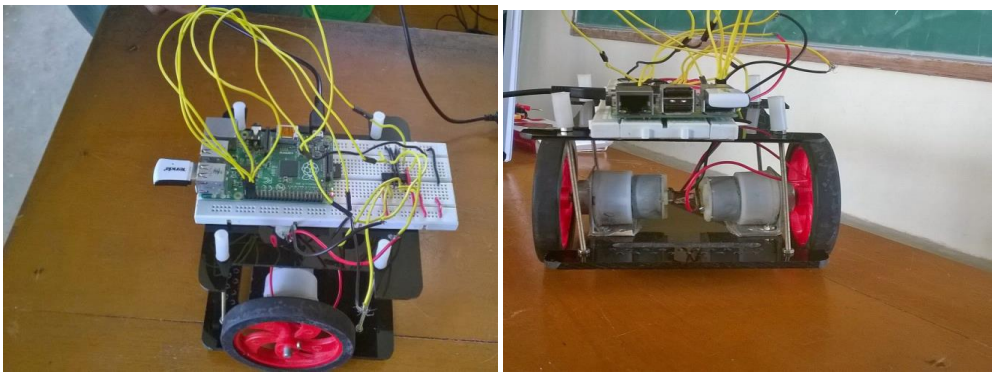

Fig 2.11 Bot connected with Raspberry Pi board through motor driver IC

Chapter 3

3. Background Theory

The fundamental theories which are used to design and develop the system are explained in this chapter.

3.1 Haar Cascade Classifier:

Haar cascade classifier uses cascade of weak and strong classifiers which uses haar like features for classification. That is if have to detect an object, for example to detect face, using cascade classifier, first we to look for the most rough feature of the face like presence of eye, if it is present then the next classifier will look for next feature likewise cascaded classifiers will play for detection of an object.

Initially a window of particular size is selected, which will slide over the entire image. The area covered at that time will be used by the classifiers for detection of a particular feature. To overcome the scaling problem, the window size will be varied over a range. To improve the speed of the process the integral image concept. Integral image is a simple 2-D matrix same as the original image which act as lookup table containing the sum of all elements located up-left of the original image. With the help of integral image one can calculate the sum of the pixels in a rectangular region with the help of four points in integral image as described in the fig 3.1 and equation 3.1

Equation 3.1:

$$Sum = I(C) - I(A) - I(B) - I(D)$$

Where, I(x) – value in the integral image at position x. ABCD – rectangular region which covers the certain foreground part of the image whose area has to be calculated

.

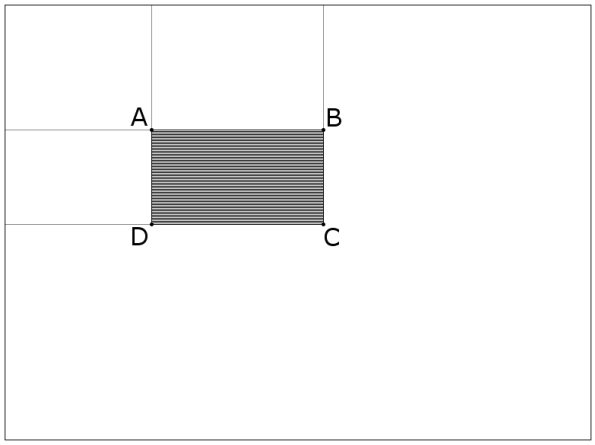

Fig 3.1Finding the sum of the shaded rectangular area

As stated above, Haar cascade classifier uses Haar like features, which is the difference of the sum of the pixels inside the white and black region the rectangle which is masked over a certain region in the image.

The masking rectangle may have n-rectangle features, where n is 2, 3 or 4, i.e, the rectangle will contain 2, 3 or 4 segments of black and white regions for detecting different features as shown in fig 3.2

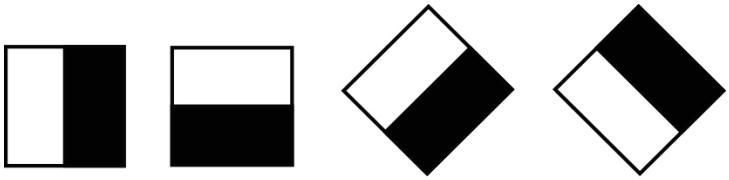

Fig 3.2 a Edge Features

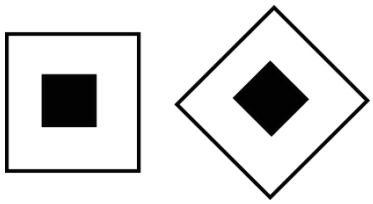

Fig 3.2 b Corner Features

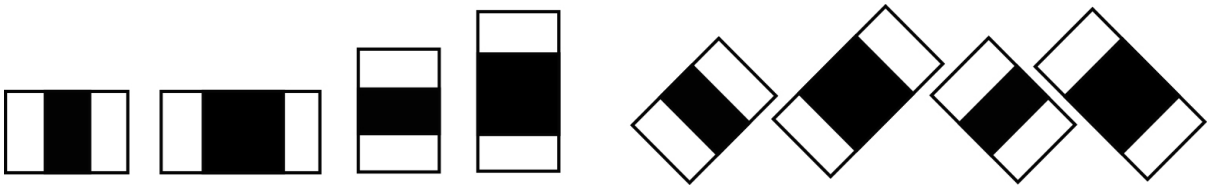

Fig 3.2 c Line Features

3.2 YCbCr Color Space

In YCbCr, Y represents the luminance component, Cb and Cr are chroma components which are blue-difference and red-difference. The advantage of using this color space it has perceptual information about the color, so it is easy to set color range to detect the objects of particular color.

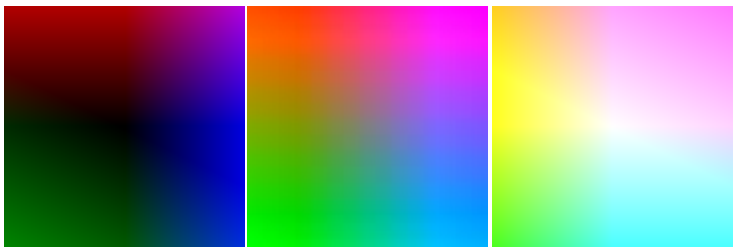

Fig 3.3 Chrominance plane for Y = 0, 128 and 255 (X-axis – Cb, Y-axis – Cr)

Luminance part will determine the brightness. If it is too low, the image will be dark and if it is high the actual color of the object will be perceived. The chrominance plane of the YCbCr color space for different luminance value is shown in Fig 3.3

As most of the capturing device gives the image in RGB color space, the conversion from RGB to required color space is needed, he following equations represents the RGB to YCbCr color space conversion,

$$Y = 16 + (65.738*R) / 256 + (129.057*G)/256 + (25.064*B)/256$$

$$Cb = 128 – (37.945*R)/256 – (74.494*G)/256 + (112.439*B)/256$$

$$Cr = 128 + (112.439*R)/256 – (94.154*G)/256 – (18.285*B)/256$$

3.3 Dilation and Erosion:

Dilation is the process of expanding the white region in the binary image. The following steps tells how to perform dilation on a given image

- Choose the shape of the structuring element, which used for finding the white neighborhood pixels.
- Move the center of the structuring element over each pixel of the image (using padding of zeros to move over the corner and edge pixels)
- If there is at least one pixel under the structuring element is white, then make the pixel coinciding with the center of structuring element as white.

The resulted image will have the expanded white borders and the gaps and dots inside the white region would be disappeared.

Erosion is the exact opposite process of dilation, where the white region in the image will shrink or the black region is expanded. The step by step procedure for erosion is given below.

- Choose the shape of the structuring element.
- Move the centre of the structuring element over each pixel of the image.
- If and only if all the pixels under the structuring element is white, then make the pixel coinciding with the centre of structuring element as white.

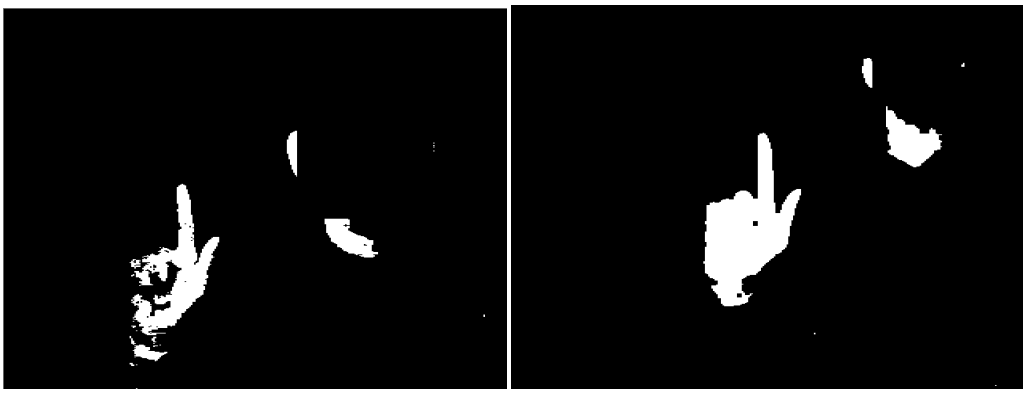

Fig 3.4 Image before (left) and after Closing operation(right)

Closing operation is process of removing gaps in a binary image. It involves dilation followed by erosion. At the first stage, as a result of dilation, the gaps in the white region will be filled, but the borders will also be expanded, if erosion is done on it the borders will be shrine.

3.4 Connected Component Labelling:

Connected component labelling is the process of segmenting the elements in the image based on connectivity i.e. all the connected pixels will be labelled with same value. For binary image, it can be done by the following method

- Scan the image pixel by pixel.

- If the scanned pixel is white, then check for the white pixel in the neighbourhood pixels for a desired connectivity (4, 8, etc).
- If any neighborhood pixel is already labelled, then label the current pixel with that label.
- If all the neighborhood pixels are not already labelled, then label the current pixel with new label value.

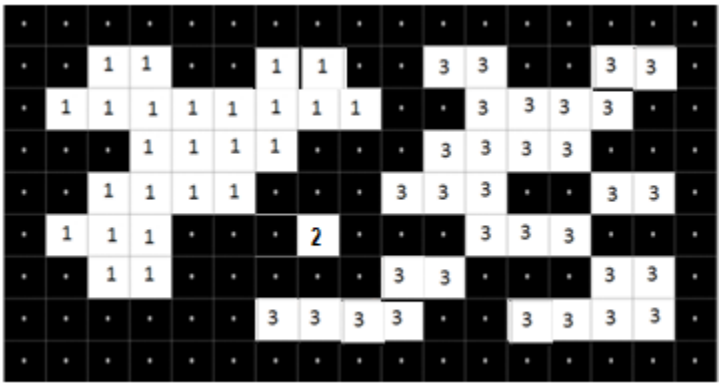

Fig 3.5 Connected components labelling

Fig 3.4 shows the connected component labelled binary image for 8 connectivity, with three islands of white region.

3.5 Contour and Convex hull:

Contour is a curve joining the continuous point along the boundary. It is similar to edge detection, where a suitable mask is convolved with the image matrix to detect the boundary, but in contour detection each pixel in the image are processed to find its continuity with the neighborhood pixels. The advantage of contour over edge detection is that the boundary of an

object detected need not be closed, while contour of a region will always be closed, so in order to operate with the shape of an object, contour is preferred.

Convex hull is the sequence of points chosen from the contour, such that points in the contour between two convex hull points will not lie outside the line drawn between the convex hull points.

Fig 3.6 Contour of the hand region and its convex hull

3.6 Orientation Calculation:

To calculate the orientation of a region in the image, a line needs to be fitted along the principle component of the region. The orientation angle is measured with respect to vertical axis with negative vale corresponds to left side rotation and positive value represents right side rotation, as shown in fig 3.7

Fig 3.7 Orientation of hand region with respect to vertical axis (Right and left side rotation)

3.7 Hu Moments:

Image moments are the weighted average of the pixel intensities, given by the equation

$$M_{ij} = \sum_i \sum_j x^i . y^j . I(x, y)$$

Where, I(x, y) is the image matrix, (x, y) are index of the image matrix, (i, j) represents the order of moments. Hu moments are the scale and rotational invariant moments, which a are normalized by the 00^{th} moment scaled by a scaling factor. The seven Hu moments are calculated using the following equations

$$I_1 = \eta_{20} + \eta_{02}$$

$$I_2 = (\eta_{20} + \eta_{02})^2 + 4.\eta_{11}^2$$

$$I_3 = (\eta_{30} - 3.\eta_{12})^2 + (3.\eta_{21} - \eta_{03})^2$$

$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$

$I_5 = (\eta_{30} - 3.\eta_{12}).(\eta_{30} + \eta_{12}).[(\eta_{30} + \eta_{12})^2 - 3.(\eta_{21} + \eta_{03})^2] + (3.\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3.(\eta_{30} + \eta_{12})^2 - (\eta_{21} - \eta_{03})^2]$

$I_6 = (\eta_{20} + \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 + 4.\eta_{11}.(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})]$

$I_7 = (3.\eta_{21} - \eta_{03}).(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3.(\eta_{21} + \eta_{03})^2] + (\eta_{30} - 3.\eta_{12}).(\eta_{21} + \eta_{03})[3.(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$

I_1 is the simply moment of inertia around the centroid of the image, it the pixel intensities are considered as physical densities. I_7 is the skew moment which will help in distinguishing mirror images.

3.8 SVM classifier:

Support Vector Machine is a classifier, which uses supervised learning algorithm for training. In case of linear classification, the SVM classifier has to find the line separating the features, this line is known decision boundary line. The decision line has to be at maximum distance from the feature sets. The distance between the decision boundary and the feature value close to it is known as margin of the classifier. If the margin is high, then the classifier will classify with high accuracy.

For example, if we have to classify the two sets of data in the image, say red and blue objects, the SVM finds a line f(x) = a*x1 + b*x2, which separates the two datasets with high margin value as shown in the figure 3.6.

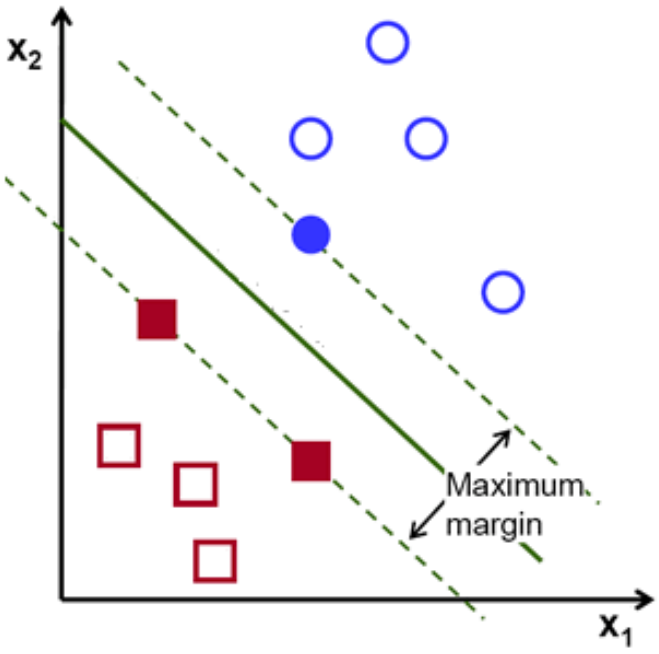

Fig 3.8 Linear SVM Classifier

SVM classifier can also classify the non-linear data by shifting the data to higher dimensions at which the dataset become linearly separable. A hyper plane is used for classifying the multi-feature dataset.

Chapter 4

4. Hardware requirements & Installation procedures

4.1 System Requirements

- **Operating system :** Windows 7/8, Ubuntu (Trusty Tahr)
- **Processor :** core i3
- **RAM:**1GB
- **Space required:**1GB

The above mentioned specifications are recommended for optimum working conditions.

4.2 Python Installation

Python is a general purpose, high level programming language. It was developed by Guido van Rossum. It became popular in very short time, mainly because of its simplicity and code readability. The most important feature of python is that, it enables the programmers to express his/her ideas in a relatively few lines of codes compared to other high level languages like C++, Java, etc. Python supports object oriented, functional or procedural paradigm. The python interpreter allows the python codes to be executed on different variety of operating systems.

To download and install the latest version of python interpreter, follow the link

i) For windows: www.python.org/downloads/windows

ii) For Linux/Unix: www.python.org/downloads/source

We have used python version 2.7.3 in our project.

4.3 Open CV – Python Binding

Open CV is a freeware library package aimed at real time computer vision applications, distributed under open-source BSD license. It was developed by Intel (Russian Research Center)

in 1999 by Gary Bradsky. Its first release came out in the year 2000. It is compatible with various programming languages like Python, Java, C++, etc. and it runs on different platforms like Linux, Windows, OS X, iOS, Android, Blackberry 10, Open BSD, etc.

Before binding Open CV with python, below python packages have to be downloaded and installed in the default python location.

- Numpy (Version 1.7.1)
- Scipy (Version 0.9.0)

Usually the default location of python is C:/Python27/.

- To download the latest version of Open CV in windows operating system, visit
  - http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.6/OpenCV-2.4.6.0.exe/download
- Extract the contents by double clicking it and copy the file "cv2.pyd" from the below mentioned folder "**opencv/build/python/2.7**".
- Paste it to "**C:/Python27/lib/site-packages"**
- To check whether it is successfully installed, type the following commands in python IDLE,

```
>>> import cv2
>>> print cv2.__version__
```

If the version is printed out, then the Open CV library is successfully installed with Python.

4.4 Camera Specifications

In our experiment, we used the built-in webcam on the laptop to capture the real time video for every frame. This webcam can provide video capture with a resolution of 640 × 480at 20 frames per second.

4.5 Raspberry Pi

It is a low cost, credit card-sized single board computer. It is developed in the United Kingdom by Raspberry Pi Foundation. It is one of the most widely used educational development kits. It can also be used as a small sized personal computer since it runs on Linux kernel based operating system and supports many applications like word processor, music player, etc. However, the system design does not include any built-in hard disk. Instead, it relies on an external SD card for the booting and long term data storage. The system does not need any heat sink or cooling system, since it does not become hot unless it is overclocked. The absence of real time clock results in the fact that it cannot keep track of the date and time when it is not powered on. There are various models of Raspberry Pi available in the market. They are named as A, A+, B, B+ and 2.0

In our project, we used a Raspberry Pi B+ model whose system specifications are as follows:

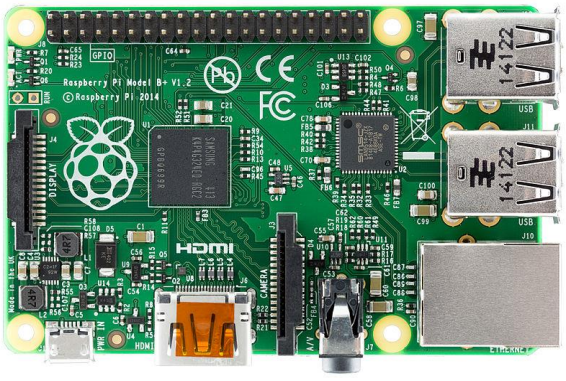

Fig. 4.1Raspberry Pi Model B+ View

4.5.1 Specifications

- **System on Chip**: Broadcom BCM2835
- **CPU**: 700 MHz Single Core ARM1176JZF-S
- **GPU**: Broadcom Video Core IV @ 250 MHz, OpenGL ES 2.0 (24 GFLOPS)
- **Memory(SDRAM)**: 512 MB (shared with GPU)
- **USB 2.0 :** 4 X USB Ports
- **Video Input**:5-pin MIPI camera interface (CSI) connector, used with the Raspberry Pi camera
- **Video Output**: HDMI, 14 resolutions from 640×350 to 1920×1200 plus various PAL and NTSC standards
- **Audio Output**: Analog via 3.5 mm phone jack; digital via HDMI
- **On-Board Network:** 10/100 Mbit/s Ethernet (8P8C) USB adapter on the third/fifth port of the USB hub
- **Power Ratings**: 600mA (3.0 W)
- **Power Source**: 5V via Micro USB
- **GPIO:** 40 pins

4.5.2Raspbian OS installation

The Raspberry Pi primarily uses Linux-kernel-based operating systems. It uses NOOBS as install manager. To install the Operating system successfully, following instructions are to be carefully performed.

- Download NOOBS OS IMAGE from http://downloads.raspberrypi.org/NOOBS_latest

- Extract the image and copy the contents in SD-CARD (it should be in FAT-32 format), minimum required size is 4GB.
- Place the SD-CARD in raspberry pi board.
- Connect pi to display using HDMI cable and connect input devices.
- Finally connect power USB cable to switch on the board.
- Select the OS to be installed (Recommended OS is Raspbian) and click install.

4.5.3 GPIO Header

The Raspberry Pi Model B+ has a 40 pin GPIO header. These pins can be configured as either general-purpose input or general-purpose output pins. Additionally, all the GPIO pins have alternative functions.

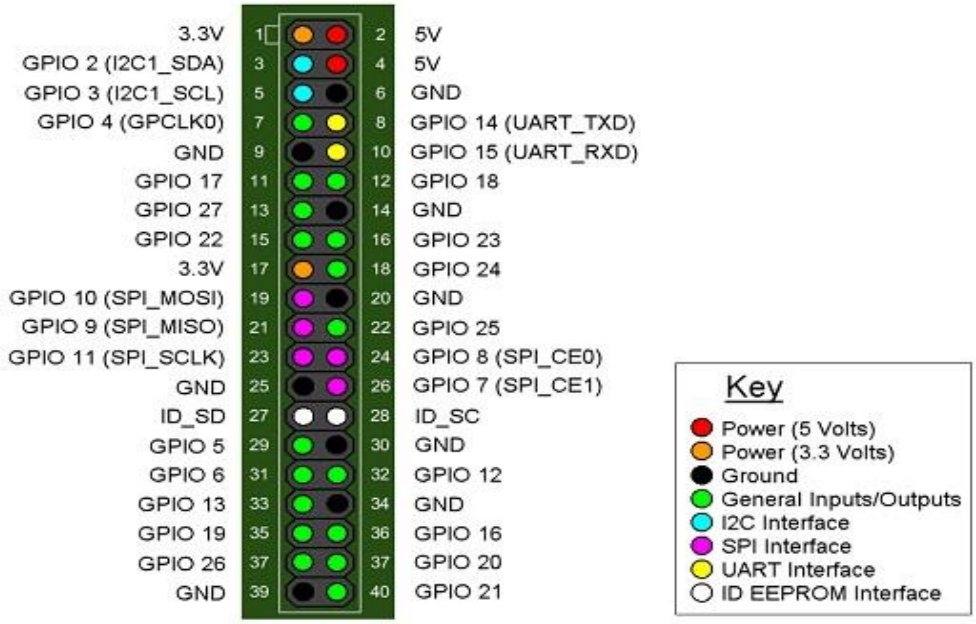

Fig 4.2 Raspberry Pi Model B+ GPIO Header

4.6Configuration of Ethernet and Wi-Fi:

4.6.1 Raspberry pi board:

- Open terminal and run "*sudonano /etc/network/interfaces*"
- In the opened configuration file add the following lines

  iface eth0 inetdhcp

  iface wlan0 inetdhcp

  psk_ssid "[WLAN network name]"

  psk_passwd "[WLAN network password]"

4.6.2 Computer or Laptop

For Windows OS based systems:

- Open command prompt terminal and run the following command "netshwlan set hostednetworkssid = [WLAN network name] key = [WLAN network password]"
- Press enter, then network will be created.
- After that, run "netshwlan start hostednetwork". The hosted network will start.
- To configure Ethernet click "open network and sharing center"
- Click "change adapter settings" → "wireless network connection" → "properties" →"sharing TAB"
- Check the "allow other network users to connect through this computer's internet connection"
- In " home networking connection" drop down box click "local area connection"0

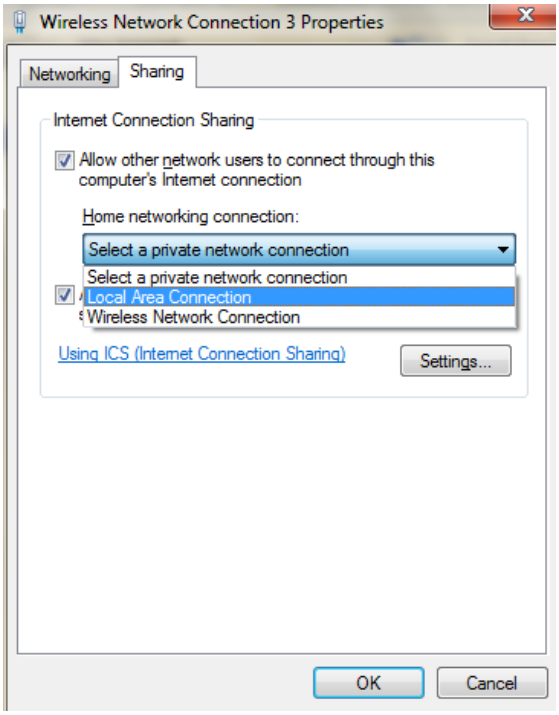

Fig 4.3 Wireless Network Connection Properties

For Linux Based System:

- Open “edit connections” and Double click “wired connections”.
- In the “IPV4 settings tab”, select method:”shared to other computers”.

4.7 Software Required:

4.7.1 Raspberry Pi Board

In Raspberry Pi Board, before executing the program, some software have to be updated and installed. It is done using the following commands

- Open terminal and run following commands:
    1. Sudo apt-get update
    2. Sudo apt-get upgrade

3. Sudo apt-get install python-scipy
4. Sudo apt-get install python-matplotlib
5. Sudo apt-get install python RPi.GPIO
6. Sudo apt-get install python-numpy

4.7.2 Computer or Laptop:

- Download putty software software from following links

  Putty: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

For Linux based systems, there is no need of any specific software.

4.8 Remote access for Raspberry Pi

4.8.1 Windows based system

- After configuring above procedure, connect Ethernet cable and wifi driver.
- In system open command prompt and run “arp –a” command
- Note the ip – address corresponding to dynamic status
- Type the ip – address in putty
- Check the ” ssh “ radio button and port no. is set to 22
- Click connect, then remote desktop will appear in the system.

4.8.2 Linux based system

- Open terminal and run ssh pi@[ip address]
- Then remote desktop will appear.

4.9 Motor Driver IC

L293D is a dual H-bridge motor driver IC. It consists of 16 pins with Dual in line package. It takes a low current control input signal and produces a high current output signal. This output signal is used to drive DC motors. It consists of two in-build H-bridge driver circuits. Hence it can be used to drive two DC motors simultaneously both in forward and reverse directions.

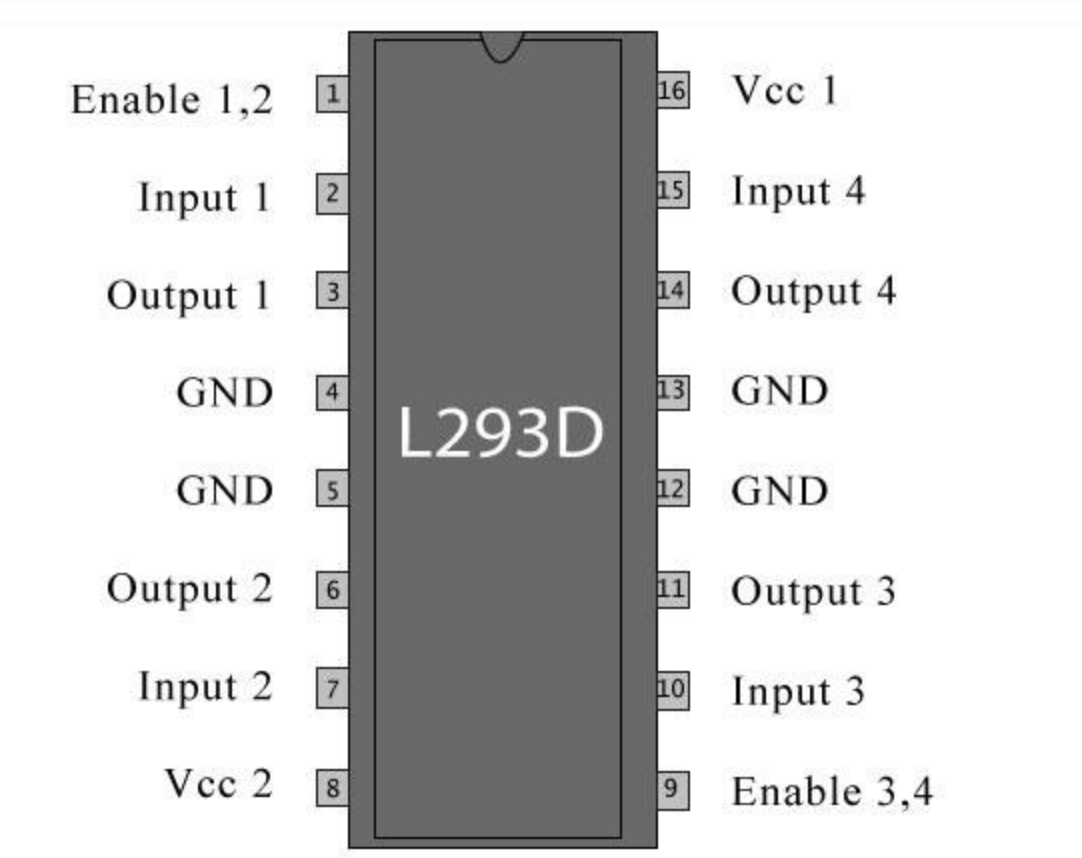

Fig 4.4 L293D Motor Driver IC

Each motor is controlled by providing input logic to pins 2 & 7 (motor 1) and 3 & 15 (motor 2) respectively. The enable pins 1 and 9 must be given logic 1 to enable the associated

driver circuits. Vcc1 pin must be given 5V D.C. power supply. The power supply for the motor is given to the pin Vcc2. The supply voltage can be given in the range of 5 V to 36 V. The output pins are connected as the inputs to the DC motor.

For controlling the motors, the input logic is given as follows.

- Pin 2 = logic 0 and Pin 7 = logic 0, idle state.
- Pin 2 = logic 0 and Pin 7 = logic 1, anti-clockwise direction.
- Pin 2 = logic 1 and Pin 7 = logic 0, clockwise direction.
- Pin 2 = logic 1 and Pin 7 = logic 1, idle state.

Similarly, for driving the second motor, input logic is given across the input pins 10 & 15.

4.10 DC motor Specifications

We have used two DC motors as a pair to use it as a car. Its specifications are as follows.

- **Output RPM**: 300 rpm
- **Input Voltage**: 9 to 12 V
- **Stall Current**: 500 to 600 mA
- **Shaft Length**: 2.4 cm

4.11 Power Sources

We have used a 12 V power supply for driving the DC motors. It is given through motor driver IC to amplify the current. The enable pins can be given a 5 V supply from Raspberry Pi power supply pins.

Chapter 5

5. Result and Analysis

5.1 Skin Color Detection:

The accuracy of skin color detection, will affect the overall performance of the system as the skin detection is at the basic level. As thresholding technique is used in this, for skin color detection, the accuracy can be improved by proper choice of thresholding values in suitable color space. In this section the result of using following color space is analyzed for the sample image shown in fig 5.1,

- RGB
- HSV
- YCbCr

Fig 5.1 Sample image used for skin color detection analysis

5.1.1 RGB Color Space:

As in RGB color space, the chrominance and luminance cannot be separated the thresholding will little complex. The thresholding condition is given by

- R > 95
- G > 40
- B > 20
- Max(R, G, B) – Min(R, G, B) > 15
- Abs(R – G) > 15
- R > G
- R > B

Where, R, G and B represents the Red, Blue and Green color components of the image pixel. For a pixel color to be a skin color it should satisfy the above all conditions. The result of skin color detection using the above the conditions on the sample image is shown in fig 5.2,

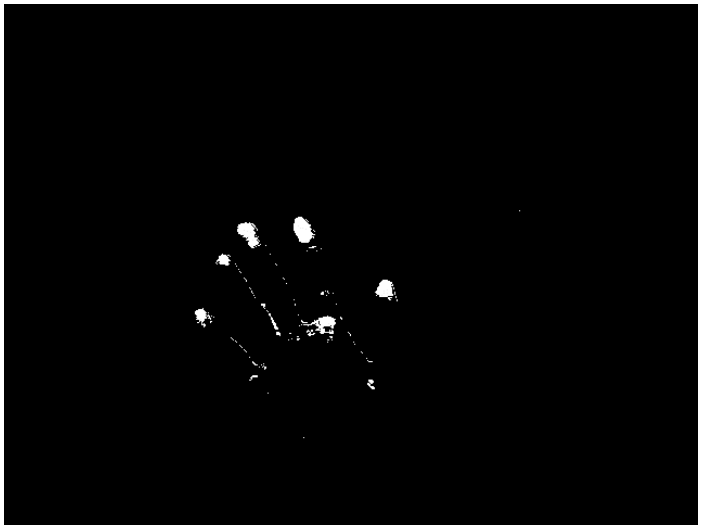

Fig 5.2 Skin color detection in RGB color space

5.1.2 HSV Color Space:

In HSV color space, chrominance part is represented by H and S. And the luminance part is represented by V. The thresholding condition for HSV color space to identify skin color pixels is given by

(H<20)*(50<S)*(S<150)*(0<V);

- H > 20
- 50 < S <150

where H and S are Hue and Saturation respectively, As V is luminance value it can be in full range, i.e. no constrained on it. The result of skin detection in HSV color space for the sample image is shown in fig 5.4

Fig 5.3 Skin color detection in HSV color space

5.1.3 YCbCr Color Space:

As in HSV color space, here also the chrominance and luminance part are separate. The chrominance part is given by Cb and Cr values. And the brightness is represented by Y value. The thresholding condition for skin color detection is given by

(135<Cr)*(Cr<183)*(120<Cb)*(Cb<154);

- 135 < Cr < 183
- 120 < Cb < 154

where, Cb and Cr are blue difference and red difference component respectively and Y is the intensity or brightness component. The result of skin color detection for the sample image is shown in fig 5.5

Fig 5.4 Skin color detection in YCbCr color space

5.2 Illumination Condition and Background Environment:

For good performance of the system, there are two main constrained on the background environment and illumination condition. If the illumination condition is not proper, then the system will not detect hand or it will detect the object which is not a hand, because of the following reasons

- Skin color object in the scene may be dark because of shadow or low light condition.
- Face in the scene may not be properly detected and subtracted because of not enough brightness condition, so skin color region of the face may be detected as hand.

The image shown in fig 5.5, shows the detection of face region also because of not proper detection of face subtraction

Fig 5.5 False detection of hand region

The presence of skin colored back ground also adversely affect the system performance because of segmentation of hand from the background is not possible at this environment.

5.3 Output:

The fig 5.6 shows the final output under the environment conditions which satisfies the constrains mentioned in section 5.2

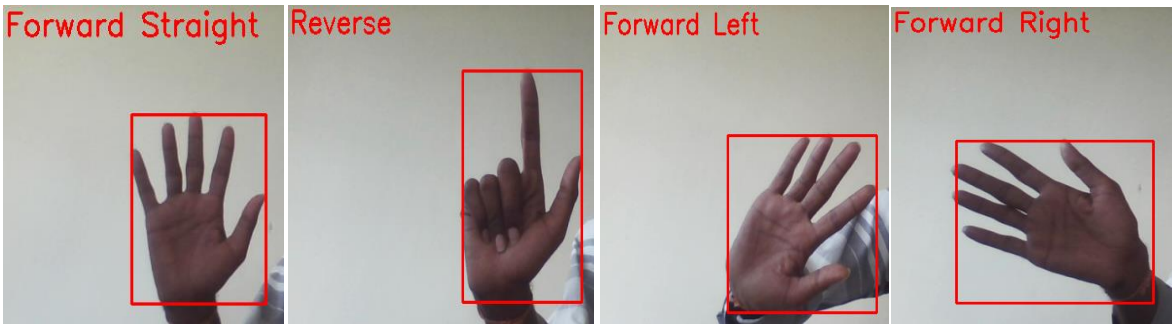

Fig 5.6 Final output

Chapter 6

6. Conclusion and Future recommendation

6.1 Conclusion:

The result of explicit skin color thresholding in different color spaces discussed in section 5.1, shows that YCbCr color space will provide the better performance in suitable environment condition. The entire system was programmed using python programming language. If more than one hand is shown in front of webcam, one which is very close to the cam will only be considered. The system can be used to control a remote device using a processing with a video capturing facility in it, but because of constrain involved in background condition and lightning condition, it cannot be used in all the environment. The overall objective of the project is completed with the satisfactory results.

6.2 Future recommendations:

The overall performance of the system can be improved if shape matching process by using contour matching algorithms is used in efficient way. The results obtained from shape matching algorithm between obtained sample image and reference image, is used to reduce the false detection.

References

[1] A. Micilotta and R. Bowden, “View-based location and tracking of body parts for visual interaction,” in Proc. BMVC, 2004.

[2] C. Schüldt, I. Laptev, and B. Caputo, “Recognizing human actions: A local SVM approach,” in Proc. ICPR, 2004.

[3] Feng Wang, Chong-Wah Ngo, “Simulating a Smartboard by Real-Time Gesture Detection in Lecture Videos”, IEEE Transactions On Multimedia, 2008.

[4] H. Zhou and T. S. Huang, “Tracking articulated hand motion with Eigen dynamics analysis,” in Proc. Int. Conf. Comput. Vis., 2003.

[5] L. Yun and Z. Peng, “An automatic hand gesture recognition system based on Viola-Jones method and SVMs,” in Proc. 2nd Int. Workshop Comput.Sci. Eng., 2009.

[6] M. Turk, “Gesture recognition,” in Handbook of Virtual Environment Technology. Mahwah, NJ: Lawrence Erlbaum., 2001.

[7] Nasser H. Dardas and Nicolas D. Georganas, “Real-Time Hand Gesture Detection and Recognition Using Bag-of-Features and Support Vector Machine Techniques”, IEEE Transactions On Instrumentation And Measurement, 2011.

[8] Qing Chen, Nicolas D. Georganas, “Hand Gesture Recognition Using Haar-Like Features and a Stochastic Context-Free Grammar”, IEEE Transactions On Instrumentation and Measurement, 2008.

[9] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2006.

[10] Y. Ivanov and A. Bobick, “Recognition of visual activities and interactions by stochastic parsing,” IEEE Trans. Pattern Anal. Mach. Intell., 2000.

[11] A. Stolcke, “Bayesian learning of probabilistic language models,” M.S. thesis, Univ. California, Berkeley, CA, 1994.

[12] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog., 2001.

[13] G. Bradski, A. Kaehler, and V. Pisarersky, “Learning-based computer vision with Intel’s open source computer vision library,” Intel Technol, 2005.

[14] J. Yao and J. R. Cooperstock, “Arm gesture detection in a classroom environment,” in Proc. IEEE Workshop Appl. Comput. Vis., 2002.

[15] C.W. Ng and S. Ranganath, “Gesture recognition via pose classification,” in Proc. 15th ICPR, 2000.